Exercise: Image and Video Processing

Sheet 1 – Image Similarity

General hints

You can use JAVA or C++ to program the following exercises (the following function declarations are specified in C++).

The functionality of **JAVA** to load, process, or save images is much higher. Use this functionality to load and save bitmap images (BMP).

For **C++**, the class ***Image*** is provided as source code to load and save images in PPM format. Class ***Image*** supports the following functionality:

`Image (int width, int height, int band)`	Create and image of width, height, and number of color channels
`Image (string filename)`	Load image from file (PPM format)
`resize (int width, int height, int band)`	Change image size
`clear (unsigned char value)`	Set all pixels to one value
`read (string filename)`	Load image
`write (string filename)`	Save image

Exercise 1 – Class ConvertColorspace

Implement a class *ConvertColorspace* to convert RGB images to YUV images and vice versa.

- Read the description of the YUV format.
- Which maximal and minimal values are possible if an RGB image is converted to YUV?
- Implement the class *ConvertColorspace* and the following methods*:*
  - `void RGB2YUV (const Image &src, Image &dest);`
  - `void YUV2RGB (const Image &src, Image &dest);`

  Add a constant value of *127.5* to U and V component to avoid negative values. Subtract this value before conversion from YUV to RGB.
- Use images ***test1***, ***test2*** and ***test3*** to test if the conversion is lossless or lossy. Convert an image into YUV and back to RGB, and compare it with the original RGB image. What is the maximum pixel difference between these two images?
- Create an image of size 256x256 and set all pixels of the first color channel to 128. Set the pixel in the second color channel to the particular column number, and the pixel of the third channel to its row number. Convert the image with *YUV2RGB* and store this YUV image to disk. How does the image look like?

Exercise 2 – Class ImageDistance

Implement the class ***ImageDistance*** and the following 3 methods to describe the difference between two images:

```
// sum of absolute difference
void SAD (Image &img1, Image &img2, double &diff);

// histogram difference
void HD (Image &img1, Image &img2, double &diff);

// standard deviation
void SD (Image &img1, Image &img2, double &diff);
```

- The histogram values in method *HD* should be stored in an array or vector. What is the dimension of the vector?
- Load the grayscale images ***test4a*** and ***test4b***, and calculate the distance values for the three distance metrics. What do you recognize?
  Hint: In grayscale images, you have to consider the first (color) channel only.
- Which kind of cut can be detected with each function?

Exercise 3 – Quality evaluation of cut detection

- What is the meaning of *precision* and *recall*? Why do we need the third measurement (*F1*)?
- Give a simple example where the *precision* is high but *F1* is low. Is it possible to find a similar example for recall?

Exercise 4 – Complexity

- Estimate the complexity of the following techniques to calculate the similarity of two images:
  - Sum of absolute pixel difference
  - Histogram
  - Standard deviation
  - Edge based contrast
  - ECR

Documentation: YUV

Source: Wikipedia, the free encyclopedia (en.wikipedia.org).

YUV is the color space used in the PAL system of television broadcasting which is the standard in most of Europe and some other places. Y stands for the luminance component (the brightness) and U and V are the chrominance (color) components. The YCbCr or YPbPr color space, used in computer component video, is derived from it (Cb/Pb and Cr/Pr are simply scaled versions of U and V), and is sometimes inaccurately called "YUV".

YUV signals are created from an original RGB (red, green and blue) source. The weighted values of R, G and B are added together to produce a single Y signal, representing the overall brightness, or luminance, of that spot. The U signal is then created by subtracting the Y from the blue signal of the original RGB, and then scaling; and V by subtracting the Y from the red, and then scaling by a different factor. This can be accomplished easily with analog circuitry.

The following equations can be used to derive *Y*, *U* and *V* from *R*, *G* and *B*:

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = 0.492(B - Y) = -0.147R - 0.289G + 0.436B$$

$$V = 0.877(R - Y) = 0.615R - 0.515G - 0.100B$$

Here, *R*, *G* and *B* are assumed to range from 0 to 1, with 0 representing the minimum intensity and 1 the maximum.

- If $[R \;\; G \;\; B]^T = [1 \;\; 1 \;\; 1]$ then $[Y \;\; U \;\; V]^T = [1 \;\; 0 \;\; 0]$. In other words, the top row coefficients sum to unity and the last two rows sum to zero.

Luminance/chrominance systems in general

The primary advantage of luminance/chrominance systems such as YUV is that they remain compatible with black and white analog television. The Y signal is essentially the same signal that would be broadcast from a normal black and white camera (with some subtle changes), and the U and V signals can simply be ignored. When used in a color setting the subtraction process is reversed, resulting in the original RGB color space.

Another advantage is that the signal in YUV can be easily manipulated to deliberately discard some information in order to reduce bandwidth. The human eye actually has fairly low color resolution: the high-resolution color images we see are processed by the visual system by combining the high-resolution black and white image with the low-resolution color image. Using this information to their advantage, standards such as NTSC reduce the amount of signal in the chrominance considerably, leaving the eye to recombine them. For instance, NTSC saves only 11% of the original blue and 30% of the original red, throwing out the rest. Since the green is already encoded in the Y signal, the resulting U and V signals are substantially smaller than they would otherwise be if the original RGB or YUV signals were sent. This filtering out of the blue and red signal is trivial to accomplish once the signal is in YUV format.

However this process, obviously, reduces the quality of the image. In the 1950s when NTSC was being created this was not a real concern because common equipment could not display images any better than the quality of the signal they were already receiving. But today a modern television can display more information than is contained in these lossy signals. This has led to a number of attempts to record images with as much of the YUV signal as possible, including S-Video on VCRs. YUV was also used as the standard format for common video compression algorithms such as MPEG-2, which is used in digital television and for DVDs. The professional CCIR 601 uncompressed digital video format also uses the YUV color space, for compatibility with previous analog video formats, which can then be easily mixed into any output format needed.

YUV is a versatile format which can be easily combined into other legacy video formats. For instance if you amplitude-modulate the U and V signals onto quadrature phases of a subcarrier you end up with a single signal called **C**, for *chroma*, which can then make the YC signal that is S-Video. If you mix the Y and C signals, you end up with composite video, which almost any television can handle. All of this modulating can be accomplished easily in low-cost circuitry, while the *demodulation* is often very difficult indeed. Leaving the signal in the original YUV format thus made DVDs very simple to construct, as they could easily down mix to support either S-video or composite and thus guarantee compability with simple circuits, while still retaining all of the original information from the source RGB signal.